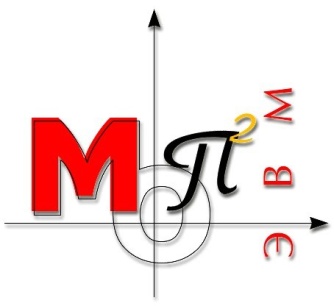
Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждения высшего образования   
«Южный федеральный университет»

Инженерно-технологическая академия

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**

Лабораторная работа №1

по дисциплине

"ООП"

на тему

"Классы и объекты в С++"

**Выполнил:**

студент группы КТбо2-6

Пустовой А.В

**Проверил:**

Таганрог, 2020

**Вариант №6**

**Задание:**

Определить класс многочленов Polinom от одной переменной, задаваемых степенью многочлена и массивом коэффициентов, с операциями вычисления значения многочлена для заданного аргумента, сложения и вычитания многочленов.

Спецификация классов:

**Polinom**

#pragma once

//polinom.h

#include <iostream>

//Спецификация класса

class Polinom

{

public:

Polinom() = default;

Polinom(int degree, double\* multipliers);

Polinom(const Polinom& source);

//Деструктор

~Polinom();

double SolvePoli(double arg) const; //Вычисление значения полинома

void SetPoli(int degree, double\* multipliers); //Ввод полинома

Polinom AddPoli(Polinom& source); //Прибавить полином

Polinom SubstractPoli(Polinom& source); //Вычесть полином

int GetDegree() const;

//Перегрузки

Polinom operator = (const Polinom&);

Polinom operator + (const Polinom&) const;

Polinom operator - (Polinom&) const;

Polinom operator - () const;

double operator [] (const int index) const;

friend std::ostream& operator << (std::ostream&, const Polinom&);

private:

int \_degree = 0;

double\* \_multipliers = nullptr;

void cpy(Polinom& target, const int degree, const double\* multiplier);

};

**Interface**

#pragma once

#include "polinom.h"

//interface.h

class Interface

{

public:

Interface();

~Interface();

void Init();

private:

void CreatePoli();

void DeletePoli(std::string name);

void Help();

void Minus();

void Plus();

int GetIndex(std::string name);

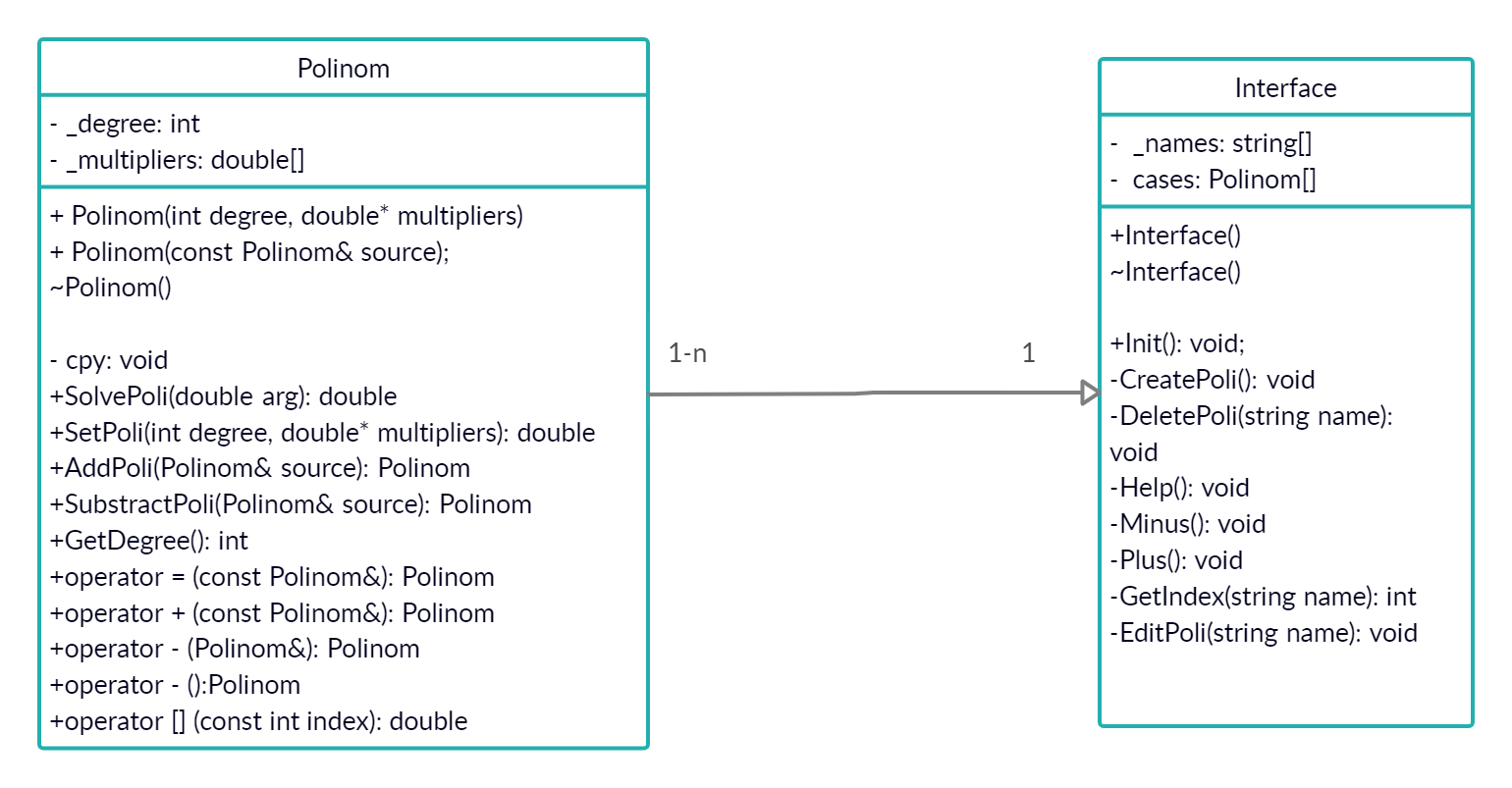
void EditPoli(std::string name);

std::string\* \_names = nullptr;

Polinom\* \_cases = nullptr;

};

**Диаграмма классов**

****

**Листинг кода**

//interface.h

#pragma once

#include "polinom.h"

class Interface

{

public:

Interface();

~Interface();

void Init();

private:

void CreatePoli();

void DeletePoli(std::string name);

void Help();

void Minus();

void Plus();

int GetIndex(std::string name);

void EditPoli(std::string name);

std::string\* \_names = nullptr;

Polinom\* \_cases = nullptr;

//interface.cpp

#define COUNT\_OF\_CASES 10

#include <iostream>

#include "interface.h"

Interface::~Interface()

{

delete[] \_names;

delete[] \_cases;

}

Interface::Interface()

{

\_names = new std::string[COUNT\_OF\_CASES];

\_cases = new Polinom[COUNT\_OF\_CASES];

}

int Interface::GetIndex(std::string name)

{

for (int i = 0; i < COUNT\_OF\_CASES; i++)

{

if (name == \_names[i])

{

return i;

}

}

return -1;

}

void Interface::Minus()

{

std::string name, name2;

std::cout << "Enter name of first polinom\n>> ";

std::cin >> name;

std::cout << "Enter name of second polinom\n>> ";

std::cin >> name2;

if (GetIndex(name) != -1 && GetIndex(name2) != -1)

{

std::cout << \_cases[GetIndex(name)] << "\n-\n";

std::cout << \_cases[GetIndex(name2)] << "\n";

std::cout << \_cases[GetIndex(name)] - \_cases[GetIndex(name2)] << "\n";

}

else

{

std::cout << "Wrond name\n";

}

}

void Interface::Plus()

{

std::string name, name2;

std::cout << "Enter name of first polinom\n>> ";

std::cin >> name;

std::cout << "Enter name of second polinom\n>> ";

std::cin >> name2;

if (GetIndex(name) != -1 && GetIndex(name2) != -1)

{

std::cout << \_cases[GetIndex(name)] << "\n+\n";

std::cout << \_cases[GetIndex(name2)] << "\n";

std::cout << \_cases[GetIndex(name)] + \_cases[GetIndex(name2)] << "\n";

}

else

{

std::cout << "Wrond name\n";

}

}

void Interface::EditPoli(std::string name)

{

int degree; double\* multipliers;

std::cout << "Enter name of polinom to edit\n>> ";

std::cin >> name;

if (GetIndex(name) != -1)

{

std::cout << "Enter new dergee\n>> ";

std::cin >> degree;

std::cout << "Enter " << degree + 1 << " new multipliers separated by a space\n>> ";

multipliers = new double[degree + 1];

for (int i = 0; i < degree + 1; i++)

{

std::cin >> multipliers[i];

}

\_cases[GetIndex(name)].SetPoli(degree, multipliers);

delete[] multipliers;

}

else

{

std::cout << "Wrong name\n";

}

}

void Interface::Help()

{

std::cout << "\"1\" - Create new polinom\n\"2\"- Delete polinom\n\"3\"- Help\n\"4\" - Set Polinom\n\"5\" - Plus\n\

\"6\" - Minus\n\"7\" - Sign inversion\n\"8\" - Solve polinom\n\"9\" - Get Degree\n\"10\"- Get multiplier\n\"11\" - Print polinom\n\"12\" - Exit\n";

}

void Interface::CreatePoli()

{

int degree; double\* multipliers; std::string name; bool isEnoughtSpace = false; int index;

std::cout << "Enter degree:\n>> ";

std::cin >> degree;

multipliers = new double[degree + 1];

std::cout << "Enter " << degree + 1 << " multipliers separated by a space\n>> ";

for (int i = 0; i < degree + 1; i++)

{

std::cin >> multipliers[i];

}

std::cout << "Enter name of polinom\n>> ";

std::cin >> name;

for (int i = 0; i < COUNT\_OF\_CASES; i++)

{

if (\_names[i].empty())

{

isEnoughtSpace = true;

index = i;

}

}

if (isEnoughtSpace)

{

Polinom temp(degree, multipliers);

\_names[index] = name;

\_cases[index] = temp;

}

else

{

std::cout << "You don't have enought space, delete something\n";

}

delete[] multipliers;

}

void Interface::DeletePoli(std::string name)

{

int index = GetIndex(name);

if (index == -1)

{

std::cout << "Can't find polinom with this name\n";

}

else

{

\_names[index] = "";

}

}

void Interface::Init()

{

int code, arg; std::string name, name2;

while (true)

{

std::cout << "Enter \"3\" - Help\n";

std::cout << ">> ";

std::cin >> code;

if (code == 1)

{

CreatePoli();

}

else if (code == 2)

{

std::cout << "Enter name of polinom to delete\n>> ";

std::cin >> name;

DeletePoli(name);

std::cout << "Complete\n";

}

else if (code == 3)

{

Help();

}

else if (code == 4)

{

EditPoli(name);

std::cout << "Complete\n";

}

else if (code == 5)

{

Plus();

}

else if (code == 6)

{

Minus();

}

else if (code == 7)

{

std::cout << "Enter name of polinom to invert\n>> ";

std::cin >> name;

if (GetIndex(name) != -1)

{

std::cout << "-(" << \_cases[GetIndex(name)] << ")" << "\n = \n" << -\_cases[GetIndex(name)] << "\n";

}

else

{

std::cout << "Wrong name\n";

}

}

else if (code == 8)

{

std::cout << "Enter name of polinom to solve\n>> ";

std::cin >> name;

if (GetIndex(name) != -1)

{

std::cout << "Enter arg of polinom\n>> ";

std::cin >> arg;

std::cout << "f(x) = " << \_cases[GetIndex(name)] << "\n" << "f(" << arg << ") = " << \_cases[GetIndex(name)].SolvePoli(arg) << "\n";

}

else

{

std::cout << "Wrong name\n";

}

}

else if (code == 9)

{

std::cout << "Enter name of polinom to get degree\n>> ";

std::cin >> name;

std::cout << "Degree = " << \_cases[GetIndex(name)].GetDegree() << "\n";

}

else if (code == 10)

{

std::cout << "Enter name of polinom to get multiplier\n>> ";

std::cin >> name;

std::cout << "Enter index of multipliers to get multiplier\n>> ";

std::cin >> arg;

if (\_cases[GetIndex(name)].GetDegree() < arg)

{

std::cout << "Wrong degree\n";

}

else

{

std::cout << "Multiplier[i] = " << \_cases[GetIndex(name)][arg] << "\n";

}

}

else if (code == 11)

{

std::cout << "Enter name of polinom to print\n>> ";

std::cin >> name;

if (GetIndex(name) != -1)

{

std::cout << \_cases[GetIndex(name)] << "\n";

}

else

{

std::cout << "Wrong name\n";

}

}

else if(code == 12)

{

break;

}

}

}

#pragma once

//polinom.h

#include <iostream>

//Спецификация класса

class Polinom

{

public:

Polinom() = default;

Polinom(int degree, double\* multipliers);

Polinom(const Polinom& source);

//Деструктор

~Polinom();

double SolvePoli(double arg) const; //Вычисление значения полинома

void SetPoli(int degree, double\* multipliers); //Ввод полинома

Polinom AddPoli(Polinom& source); //Прибавить полином

Polinom SubstractPoli(Polinom& source); //Вычесть полином

int GetDegree() const;

//Перегрузки

Polinom operator = (const Polinom&);

Polinom operator + (const Polinom&) const;

Polinom operator - (Polinom&) const;

Polinom operator - () const;

double operator [] (const int index) const;

friend std::ostream& operator << (std::ostream&, const Polinom&);

private:

int \_degree = 0;

double\* \_multipliers = nullptr;

void cpy(Polinom& target, const int degree, const double\* multiplier);

};

//polinom.cpp

#include <iostream>

#include "polinom.h"

//Определение конструкторов и методов

Polinom::Polinom(int degree, double\* multipliers)

{

\_multipliers = new double[degree + 1];

cpy(\*this, degree, multipliers);

}

Polinom::Polinom(const Polinom& source)

{

cpy(\*this, source.\_degree, source.\_multipliers);

}

double Polinom::SolvePoli(double arg) const

{

double answ = 0;

for (int i = \_degree, j = 0; i >= 0; i--, j++)

{

answ += \_multipliers[j] \* pow(arg, i); //exp(b\*log(a))

}

return answ;

}

void Polinom::SetPoli(int degree, double\* multipliers)

{

cpy(\*this, degree, multipliers);

}

Polinom Polinom::AddPoli(Polinom& source)

{

\*this = \*this + source;

return \*this;

}

Polinom Polinom::SubstractPoli(Polinom& source)

{

\*this = \*this - source;

return \*this;

}

int Polinom::GetDegree() const

{

return \_degree;

}

std::ostream& operator << (std::ostream& out, const Polinom& poli)

{

for (int i = poli.\_degree, j = 0; i >= 0; i--, j++)

{

if (poli.\_multipliers[j])

{

out << poli.\_multipliers[j] << "\*x^" << i;

if (i)

{

out << " + ";

}

}

else if (!i)

{

out << poli.\_multipliers[j];

}

}

return out;

}

Polinom Polinom::operator - () const

{

Polinom result;

result.\_degree = \_degree;

result.\_multipliers = new double[result.\_degree + 1];

for (int i = 0; i <= \_degree; i++)

{

result.\_multipliers[i] = -\_multipliers[i];

}

return result;

}

Polinom Polinom::operator + (const Polinom& source) const

{

Polinom result;

int delDegree = abs(\_degree - source.\_degree);

result.\_degree = (\_degree > source.\_degree) ? \_degree : source.\_degree;

result.\_multipliers = new double[result.\_degree + 1];

if (\_degree > source.\_degree)

{

memcpy(result.\_multipliers, \_multipliers, sizeof(double) \* (\_degree + 1));

for (int i = delDegree, j = 0; i <= result.\_degree; i++, j++)

{

result.\_multipliers[i] += source.\_multipliers[j];

}

}

else

{

memcpy(result.\_multipliers, source.\_multipliers, sizeof(double) \* (source.\_degree + 1));

for (int i = delDegree, j = 0; i <= result.\_degree; i++, j++)

{

result.\_multipliers[i] += \_multipliers[j];

}

}

return result;

}

Polinom Polinom::operator - (Polinom& source) const

{

return \*this + (-source);

}

void Polinom::cpy(Polinom& target, const int degree, const double\* multipliers)

{

target.\_degree = degree;

if (target.\_degree != degree || target.\_multipliers == nullptr)

{

delete[] target.\_multipliers;

target.\_multipliers = new double[target.\_degree + 1];

}

memcpy(target.\_multipliers, multipliers, sizeof(double) \* (target.\_degree + 1));

}

double Polinom::operator [] (const int index) const

{

if (index > \_degree)

{

throw std::out\_of\_range("OUT\_OF\_RANGE");

}

else

{

return \_multipliers[index];

}

}

Polinom Polinom::operator =(const Polinom& source)

{

cpy(\*this, source.\_degree, source.\_multipliers);

return \*this;

}

Polinom::~Polinom()

{

delete[] \_multipliers;

}